tica (depth < Plac

: = inside / : it = nt / nc, dr os2t = 1.0f - ... 3, N); 3)

st a = nt - nc, b = ntst Tr = 1 - (R0 + (1 - 1))Tr) R = (0 - nnt - 1)

= diffuse = true;

-:fl + refr)) && (depth is HADDEE

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; adiance = SampleLight(@rand I = 1) =x + radiance.y + radiance.z) = 0

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Puncture st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 100

andom walk - done properly, closely following a /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Dof urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2017

Lecture 3: "Ray Tracing"

Welcome!



tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, body pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



Ray Tracing

tic: ⊾(depth < PA

: = inside / 1 it = nt / nc, ddo ss2t = 1.0f = ont 5, N); 3)

st a = nt - hc, b + nt - hcst Tr = 1 - (R0 + (1 - 1))Tr) R = (0 * nnt - 1)

= diffuse; = true;

efl + refr)) && (depth < HAADIIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = Surviv estimation - do if; -adiance = Sampl e.x + radiance.y

w = true; at brdfPdf = Eva at3 factor = dif at weight = Mis2 at cosThetaOut = E = ((weight

andom walk - don /ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

PART 1: Introduction & shading (today)

PART 2: Reflections, refraction, absorption (next week)

PART 3: Path Tracing (later)







Ray Tracing

Ray Tracing:

World space

- Geometry
- Eye
 - Screen plane
 - Screen pixels
 - Primary rays
 - Intersections
 - Point light
 - Shadow rays
- survive = SurvivalProbability estimation - doing it proposed #f; radiance = SampleLight(%rand e.x + radiance.y + radiance.z)
- w = true; at brdfPdf = EvaluateDiffuse(L, Extension rays at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L) E * ((weight * cosThetaOut)Light transport
- andom walk done properly, closely following : /ive)

), N);

= true;

AXDEPTH)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, ppd prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





Ray Tracing

Ray Tracing:

World space

Geometry

Eye

•

), N);

= true;

AXDEPTH)

- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays
- survive = SurvivalProbability estimation - doing it prop #f; radiance = SampleLight(& rand e.x + radiance.y + radiance.z)
- w = true; at brdfPdf = EvaluateDiffuse(L, || Extension rays st3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf)) at cosThetaOut = dot(N, L) Light transport E * ((weight * cosThetaOut)Light transport
- andom walk done properly, closely following : /ive)
- ; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, ppd prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







Ray Tracing

Ray Tracing:

World space

- Geometry
- Eye
 - Screen plane
 - Screen pixels
 - Primary rays
 - Intersections
 - Point light
 - Shadow rays
- survive = SurvivalProbability Light transport adiance = SampleLight(&ra e.x + radiance.y + radiance.z
- Extension rays t brdfPdf = EvaluateDiffuse() st3 factor = diffuse = INVPI at weight = Mis2(directPdf, brdfPdf et cosThetaOut = dot(N, L) E • ((weight = cosThetaOut)Light transport
- andom walk done properly, closely foll vive)

), N);

= true;

AXDEPTH)

if:

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, A urvive; pdf; i = E * brdf * (dot(N, R) / pdf); sion = true:





We are calculating light transport backwards.





Ray Tracing

la: ⊾(depth (): = = inside /

it = nt / nc, os2t = 1.0f), N); 3)

at a = nt - nc, at Tr = 1 - (R0 Tr) R = (D * nnt

= diffuse = true;

. **:fl + refr**)) && (dep:

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProb estimation - doing it if; adiance = SampleLight e.x + radiance.y + rad

v = true; at brdfPdf = EvaluateD st3 factor = diffuse " st weight = Mis2(dire at cosThetaOut = dot(E * ((weight * cosTheta)

andom walk - done properly, /ive)

; t3 brdf = SampleDiffuse(diffuse, N, r) rvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;



لاندان مانند عليه اسط مستوغيره فلان هذا الشطح يقط سط برض عنقطة تكلابة من لريقط احلحل بن سر فلكن ذلك الخط متعن الفصل المشترك بين هذا السط وبين ط قط ق حط مبش فلات هذا السط نياس سيط م على فتط ت غنط م شيط ترقط قت د على نقطة مت وكذلك خط متع و فل محال فلا ياش ببط مت على نقطة مت سطح مستو غير سطح مستو في من ا







Ray Tracing



fl + refr)) && (), N); efl * E * diffus

AXDEPTH)

survive = SurvivalProl
estimation - doing if
if;
adiance = SampleLight
e.x + radiance.y + r

v = true; at brdfPdf = EvaluateD at3 factor = diffuse * at weight = Mis2(dire at cosThetaOut = dot(E * ((weight * cosThetaOut))

andom walk - done properly, closely follow: /ive)

st3 brdf = SampleDiffuse(diffuse, N, r1, r2, IR, urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;









Ray Tracing

tice ≰ (depth < NAS

= inside / 1 it = nt / nc, dde -552t = 1.0f - nnt -5, N); 3)

st a = nt - nc, b = nt - ncst Tr = 1 - (80 + (1 - 1)) Tr) R = (D * nnt - N * 1)

E = diffuse; = true;

-: :fl + refr)) && (depth k HAA

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(&rand I .x + radiance.y + radiance.r) > 0) &

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Ps at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely followin /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, sr urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Physical basis

Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

It does so by finding light transport paths:

- From the 'eye'
- Through a pixel
- Via scene surfaces
- To one or more light sources.

At each surface, the light is modulated. The final value is deposited at the pixel (simulating reception by a sensor).

T. Whitted, "An Improved Illumination Model for Shaded Display", ACM, 1980.



tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, body pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



Intersections

tic: k (depth < 100

: = inside / i it = nt / nc, ddo os2t = 1.0f - nn: 0, N); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R8 + (1 - - - -Fr) R = (D * nnt - N - - - -

E * diffuse; = true;

: :fl + refr)) && (depth is MAXDIII

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I .x + radiance.y + radiance.z) _____

v = true;
v = brdfpdf

it brdfPdf = EvaluateDiffuse(L, N.) Power it3 factor = diffuse * INVPI; ot weight = Mis2(directPdf, brdfPdf); it cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * .

andom walk - done properly, closely following . /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Lor pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Ray definition

A ray is an infinite line with a start point:

 $p(t) = 0 + t\vec{D}$, where t > 0.

struct Ray

};

float3 0; // ray origin
float3 D; // ray direction
float t; // distance

The ray direction is generally *normalized*.



tice k (depth < 100⊂

= inside / 1 it = nt / nc, dde -552t = 1:0f - nnt -5, N); 3)

st a = nt - nc, b = nt - ncst Tr = 1 - (R0 + (1 - 1)) Tr) R = (D * nnt - N - 1)

E * diffuse; = true;

: efl + refr)) && (depth k HAAD

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly if; radiance = SampleLight(&rand, I, II, II) e.x + radiance.y + radiance.z) > 0) III

v = true; at brdfPdf = EvaluateDiffuse(

st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * //

andom walk - done properly, closely following : /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, F1, F2, R, F3, pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Ray setup

A ray is initially shot through a pixel on the screen plane. The screen plane is defined in world space:

Camera position: $\vec{E} = (0,0,0)$ View direction: \vec{V} Screen center: $C = E + d\vec{V}$ Screen corners: $p_0 = C + ((-1, -1, 0)), p_1 = C + ((1, -1, 0)), p_2 = C + ((-1, 1, 0))$

From here:

- Change FOV by altering *d*;
 - Transform camera by multiplying E, p_0 , p_1 , p_2 with the camera matrix.



Only if $\vec{V} = (0,0,1)$ of course.

tic: ⊾ (depth ⊂ 10.

= inside / 1 + it = nt / nc, dde ss2t = 1.0f - nnt -), N); 3)

st a = nt - nc, b - nt - stst Tr = 1 - (80 + (1) Tr) R = (D * nnt - N *)

= diffuse; = true;

efl + refr)) && (depth & MANDISIN

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(%rand, I e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pours) st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

sndom walk - done properly, closely following vive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, so pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Ray setup

Point on the screen:

 $p(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0), \ u,v \in [0,1)$

Ray direction (before normalization):

 $\vec{D} = p(u, v) - E$

Ray origin:

O = E





tic: K (depth < 10

= inside / 1 ht = nt / nc, dde bs2t = 1.0f - nnt b, N); b)

= diffuse; = true;

-:fl + refr)) && (depth k HAA

D, N); -efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly ff; radiance = SampleLight(&rand, I, M, A e.x + radiance.y + radiance.z) = 0

v = true; at brdfPdf = EvaluateDiffuse(L, N) * st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directP

andom walk - done properly, closely fellow vive)

, H33 brdf = SampleDiffuse(diffuse, N, F1, F2, RR, F5 prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Ray intersection

Given a ray $p(t) = 0 + t\vec{D}$, we determine the smallest intersection distance *t* by intersecting the ray with each of the primitives in the scene.

Ray / plane intersection:

Plane:
$$\mathbf{p} \cdot \vec{N} + d = 0$$

Ray: $p(t) = 0 + t\vec{D}$

Substituting for p(t), we get

 $\begin{pmatrix} 0+t\vec{D} \end{pmatrix} \cdot \vec{N} + d = 0$ $t = -(0 \cdot \vec{N} + d) / (\vec{D} \cdot \vec{N})$ $P = 0 + t\vec{D}$





), N); -efl * E * diffuse;

AXDEPTH)

survive = SurvivalProbability(dif if; adiance = SampleLight(&rand, I, 1) e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(L t3 factor = diffuse * INVPI: at weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely f vive)

at3 brdf = SampleDiffuse(diffuse, N, r1 urvive; pdf;

1 = E * brdf * (dot(N, R) / pdf); sion = true:

Ray intersection

Ray / sphere intersection:

Sphere: $(p - C) \cdot (p - C) - r^2 = 0$ Ray: $p(t) = 0 + t\vec{D}$

Substituting for p(t), we get

$$(0 + t\vec{D} - C) \cdot (0 + t\vec{D} - C) - r^2 = 0 \vec{D} \cdot \vec{D} t^2 + 2\vec{D} \cdot (0 - C) t + (0 - C)^2 - r^2 = 0$$

$$ax^2 + bx + c = 0 \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

 $a = \vec{D} \cdot \vec{D}$ $b = 2\vec{D} \cdot (O - C)$

 $c = (0 - C) \cdot (0 - C) - r^2$

Negative: no intersections

E

 p_1



 p_0

 p_2

tica ⊾ (depth ⊂ 100

: = inside / l it = nt / nc, dde os2t = l.0f - nnt -D, N); B)

st a = nt - nc, b - nt st Tr = 1 - (R0 + (1 - 1 Fr) R = (D * nnt - N

E * diffuse; = true;

-:fl + refr)) && (depth < NAA

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly if; radiance = SampleLight(%rand, I, I, I) e.x + radiance.y + radiance.z) > 0) %%

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pour st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follo /ive)

```
;
pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, brd
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true:
```

Ray Intersection

Efficient ray / sphere intersection:

void Sphere::IntersectSphere(Ray ray)

vec3 c = this.pos - ray.0; float t = dot(c, ray.D); vec3 q = c - t * ray.D; float p² = dot(q, q); if (p² > sphere.r²) return; t -= sqrt(sphere.r² - p²); if ((t < ray.t) && (t > 0)) ray.t = t; // or: ray.t = min(ray.t, max(0, t));

Note:

This only works for rays that start outside the sphere.





tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, b) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



tic: k (depth < 12

= inside / 1 nt = nt / nc. dda 552t = 1.8f - nn 5, N); 3)

at a = nt - nc, b - nt - at Tr = 1 - (R0 + (1 - 1) Tr) R = (D * nnt - N * -

= diffuse = true;

-:fl + refr)) && (depth is Haad

D, N); refl * E * diffu: = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly if; adiance = SampleLight(%rand, I, M) e.x + radiance.y + radiance.z) > 0) %

v = true;

st brdfPdf = EvaluateDiffuse(L, N) Paurol st3 factor = diffuse INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, brd pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





lc: k (depth < 10.⊂

: = inside / l ht = nt / nc, ddo os2t = 1.0f - nnt -D, N); B)

st a = nt - nc, b - nt st Tr = 1 - (80 + (1 - 1) Tr) R = (D * nnt - N

= diffuse; = true;

: :fl + refr)) && (depth & Hood)

), N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property ff; radiance = SampleLight(%rand, I e.x + radiance.y + radiance.z) > 0) %

v = true; t brdfPdf = EvaluateDiffuse(L, N.) Provident st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 1

andom walk - done properly, closely following : /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Up) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

The End

We used *primary rays* to find the *primary intersection* point.

Determining light transport:

- Sum illumination from all light sources
- ...If they are *visible*.

We used a primary ray to find the object visible through a pixel: Now we will use a *shadow ray* to determine visibility of a light source.



tice ≰ (depth < RAS

: = inside / 1 it = nt / nc, dde os2t = 1.0f - nnt o, N); 3)

st a = nt - nc, b = ntst Tr = 1 - (R0 + (1 - 1))Tr) R = (0 - nnt - 1)

E = diffuse; = true;

efl + refr)) && (depth k MAADI

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property ff; radiance = SampleLight(&rand, I e.x + radiance.y + radiance.z) = 0)

v = true; t brdfPdf = EvaluateDiffuse(L, N.) Provident st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 10

andom walk - done properly, closely following : /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, sr urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Shadow Ray

Constructing the shadow ray:

 $p(t) = 0 + t\vec{D}$

Ray origin: the primary intersection point *I*.

Ray direction: $P_{light} - I$ (normalized)

Restrictions on *t*: $0 < t < ||P_{light} - I||$





tice k (depth < 100⊂

: = inside / l ht = nt / nc, ddo os2t = 1.0f - nnt -), N); 3)

st $a = nt - nc_1 b - nt$ st Tr = 1 - (R0 + (1))Tr) R = (0 * nnt - N)

E = diffuse; = true;

•: •fl + refr\\ 88 /death / HAVIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * P st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely fo /ive)

, H33 brdf = SampleDiffuse(diffuse, N, rl, r2, N, por prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Shadow Ray

Direction of the shadow ray: $\frac{P_{light}-I}{||P_{light}-I||}$

Equally valid:

Note that we get different intersection points depending on the direction of the shadow ray.

It doesn't matter: the shadow ray is used to determine *if* there is an occluder, not *where*.

This has two consequences:

- 1. We need a dedicated shadow ray query;
- 2. Shadow ray queries are (on average) twice as fast. (why?)

I-P_{light}

 $||P_{light} - I||$

or

I-P_{light}

I-P_{liaht}





tic: K (depth < NAS

= inside / 1 it = nt / nc, dde ss2t = 1.0f = ont), N); 3)

st a = nt - nc, b = nt - ncst Tr = 1 - (R0 + (1 - 0))Tr) R = (0 * nnt - 0)

E = diffuse; = true;

efl + refr)) && (depth k HADDII

D, N); refl * E * diffuse; = true;

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pour st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following: /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, pd4 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Shadow Ray

"In theory, theory and practice are the same. In practice, they are not."

Problem 1:

Our shadow ray queries report intersections at $t = \sim 0$. Why?

Cause: the shadow ray sometimes finds the surface it originated from as an occluder, resulting in *shadow acne*.

Fix: offset the origin by 'epsilon' times the shadow ray direction.

Note: don't forget to reduce t_{max} by epsilon.



tice ≰ (depth < 10.5

: = inside / l it = nt / nc, dde os2t = 1.0f - nnt), N); 3)

st a = nt - nc, b = nt - ncst Tr = 1 - (R0 + (1 - 0))Tr) R = (0 + nnt - 0)

= diffuse; = true;

-: :fl + refr)) && (depth & HANDIII

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly, if; radiance = SampleLight(&rand, I, &... e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Punct st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

indom walk - done properly, closely following -/ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, 1997 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Shadow Ray

"In theory, theory and practice are the same. In practice, they are not."

Problem 2:

Our shadow ray queries report intersections at $t = t_{max}$. Why?

Cause: when firing shadow rays from the light source, they may find the surface that we are trying to shade.

Fix: reduce t_{max} by 2 * *epsilon*.



fic: ⊾ (depth ∈ HASC

= inside / 1 it = nt / nc, dde -552t = 1.0f - nnt -3, N); 3)

st a = nt - nc, b = nt - nc, b = nt - stst Tr = 1 - (R0 + (1 - Tr)) R = (0 + nnt - N - T)

= diffuse; = true;

efl + refr)) && (depth < HARDITT

D, N); refl * E * diffuse; = true;

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pourch st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 000

andom walk - done properly, closely following a /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, brd pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Shadow Ray

"The most expensive shadow rays are those that do not find an intersection."

Why?

(because those rays tested every primitive before concluding that there was no occlusion)



tic: k (depth < 000⊂

= inside / 1 it = nt / nc, dde -552t = 1.0f - nnt -5, N); 3)

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 11 Tr) R = (D * nnt - N

= diffuse; = true;

-:fl + refr)) && (depth K HAND

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; adiance = SampleLight(%rand, I, Market e.x + radiance.y + radiance.z) > 0) %

w = true; st brdfPdf = EvaluateDiffuse(L, N) Promote st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

indom walk - done properly, closely following :
/ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Spin urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Transport

The amount of energy travelling from the light via the surface point to the eye depends on:

- The brightness of the light source
- The distance of the light source to the surface point
- Absorption at the surface point
- The angle of incidence of the light energy



tic: k (depth < 155

: = inside / l it = nt / nc, dde os2t = 1.0f = ont), N); 3)

st a = nt - nc, b = nt - ncst Tr = 1 - (R0 + (1 - 0))Tr) R = (D + nnt - N - 0)

= diffuse; = true;

-:fl + refr)) && (depth & HADDIII

D, N); refl * E * diffuse; = true;

AXDEPTH)

v = true; t brdfPdf = EvaluateDiffuse(L, N.) * Pour st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following : /ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, l) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Transport

Brightness of the light source:

Expressed in *watt (W)*, or *joule per second (J/s or Js*⁻¹).

Energy is transported by photons.

Photon energy depends on wavelength; energy for a 'yellow' photon is $\sim 3.31 \cdot 10^{-19}$ J.

A 100W light bulb thus emits $\sim 3.0 \cdot 10^{21}$ photons per second.



Shading

tic: ⊾(depth ∈ NA

: = inside / l it = nt / nc, dde os2t = 1.0f - nn: D, N); D)

st a = nt - nc, b - nt st Tr = 1 - (80 + (1 fr) R = (D * nnt - N -

= diffuse; = true;

efl + refr)) && (depth K HANDER

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

v = true; t brdfPdf = EvaluateDiffuse(L, N) st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPd

andom walk - done properly, closely following /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Dot prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Transport

Energy at distance *r*:

For a point light, a brief pulse of light energy spreads out as a growing sphere. The energy is distributed over the surface of this sphere.

It is therefore proportional to the inverse area of the sphere at distance *r*, i.e.:

$$E/m^2 = E_{light} \frac{1}{4\pi r^2}$$

Light energy thus dissipates at a rate of $\frac{1}{r^2}$. This is referred to as *distance attenuation*.



tice ≰ (depth < 10.5

: = inside / l it = nt / nc, dde os2t = 1.0f - nnt), N); 3)

st a = nt - nc, b - nt st Tr = 1 - (80 + (1 Tr) R = (0 * nnt - n

= diffuse; = true;

: :fl + refr)) && (depth k HADIIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pours) st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following . /ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Lor urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Transport

Absorption:

Most materials absorb light energy. The wavelengths that are not fully absorbed define the 'color' of a material.

The reflected light is thus:

 $E_{reflected} = E_{incoming} \cdot C_{material}$

Note that $C_{material}$ cannot exceed 1; the reflected light is never *more* than the incoming light.



tic: k (depth < 100)

= inside / 1 it = nt / nc, ddo os2t = 1.0f - nnt o, N); 0)

st a = nt - nc, b - nt st Tr = 1 - (R0 + (1 fr) R = (D * nnt - N *

= diffuse = true;

efl + refr)) && (depth is HADDII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(different estimation - doing it properly if; radiance = SampleLight(&rand, I, L, L) e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pours st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

sndom walk - done properly, closely following
vive)

; ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Upd) prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Transport

Energy arriving at an angle:

A small bundle of light arriving at a surface affects a larger area than the cross-sectional area of the bundle.

Per m^2 , the surface thus receives less energy. The remaining energy is proportional to:

 $\cos \alpha$ or: $\vec{N} \cdot \vec{L}$.



tice ⊾ (depth < 1920

: = inside / 1 it = nt / nc, dde os2t = 1.0f - nnt 5, N); 3)

at a = nt - nc, b - nt at Tr = 1 - (R0 + (1 fr) R = (0 ° nnt - N

= diffuse; = true;

-:fl + refr)) && (depth & HADD

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly if; adiance = SampleLight(\$rand, I, I, e.x + radiance.y + radiance.z) > 0) %

w = true; t brdfPdf = EvaluateDiffuse(L, N) Promote st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

indom walk - done properly, closely following -/ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, D) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Transport

All factors:

- Emitted light : defined as RGB color, floating point
- Distance attenuation: $\frac{1}{r^2}$
- Absorption, modulate by material color
- N dot L





tic: ⊾(depth < N4

= inside / 1 it = nt / nc, ddo ss2t = 1.8f - nnt 5, N); 3)

st a = nt - nc, b = nt - nc, st Tr = 1 - (R0 + 1)Tr) R = (D + nnt - N + 1)

= diffus: = true;

• •fl + refr)) 88 (death o HADD)

>, N); -efl * E * dif = true;

AXDEPTH)

survive = SurvivalProbability different estimation - doing it property if; radiance = SampleLight(&rand I .x + radiance.y + radiance.r) > 0) &

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pur st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) *

andom walk - done properly, closely following b /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, rl, r2, UR, Upd) prvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, b) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



Assignment 2

tic: K (depth < 10.5

: = inside / 1 it = nt / nc, ddo os2t = 1.0f = nnt), N); 3)

st a = nt - nc, b = nt + cst Tr = 1 - (R0 + (1 - 0))Tr) R = (D - nnt - N - 0)

= diffuse; = true;

-:fl + refr)) && (depth is HAAD

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(&rand, I = 0, e.x + radiance.y + radiance.r) > 0)

v = true; st brdfPdf

at brdfPdf = EvaluateDiffuse(L, N) Pauloust st3 factor = diffuse = INVPI; bt weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) = ();

andom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, bord urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Deadline assignment 1:

Wednesday May 10th, 23.59

Assignment 2: "Write a basic ray tracer."

- Using the template
- In a 1024x512 window
- Two views, each 512x512
- Left view: 3D
- Right view: 2D slice





Assignment 2

tice k (depth < 10.)

: = inside / 1 it = nt / nc, dde os2t = 1.0f - nnt -O, N); B)

at a = nt - nc, b - nt at Tr = 1 - (R0 + 1 r) R = (D * nnt - N * 1

= diffuse; = true;

-:fl + refr)) && (depth < NADET

D, N); =efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(%rand, I, L) e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPd

andom walk - done properly, closely folio /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Dp3 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Assignment 2: "Write a basic ray tracer."

Steps:

- 1. Create a Camera class; default: position (0,0,0), looking at (0,0,-1).
- 2. Create a Ray class
- 3. Create a Primitive class and derive from it a Sphere and a Plane class
- 4. Add code to the Camera class to create a primary ray for each pixel
- 5. Implement Intersect methods for the primitives
- 6. Per pixel, find the nearest intersection and plot a pixel
- 7. Add controls to move and rotate the camera
- 8. Add a checkerboard pattern to the floor plane.

9. Add reflections and shadow rays (next lecture).



For y = 0, visualize every 10^{th} ray

Visualize the intersection points



Assignment 2

ic: k (depth < Name

- : = inside / 1 ht = nt / nc, dde os2t = 1.0f - nnt 0, N); 3)
- st a = nt nc, b nt + st Tr = 1 - (R0 + (1 - 10 fr) R = (D * nnt - N - 10
- E = diffuse; = true;
- efl + refr)) && (depth k HANDES
- D, N); refl * E * diffuse; = true;
- AXDEPTH)
- survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(&rand, I, b) e.x + radiance.y + radiance.z) > 0) # 1
- v = true; at brdfPdf = EvaluateDiffuse(L, N) * F at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf
- andom walk done properly, closely follo /ive)
- ; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Spr urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Extra points:

- Add additional primitives, e.g.:
 - Triangle, quad, box
 - Torus, cylinder
 - Fractal
- Add textures to all primitives
 - Add a sky dome
- Add refraction and absorption (next lecture)
 - One extra point for the fastest ray tracer
 - One extra point for the smallest ray tracer meeting the minimum requirements.







Assignment 2

tic: ⊾ (depth < Pusson = ∗ inside / 1

- nt = nt / nc, ddo -552t = 1.0f - nnt -5, N); 8)
- st a = nt nc, b = nt ncst Tr = 1 - (R0 + (1 - 0) Tr) R = (D * nnt - N -
- = diffuse; = true;
- -:fl + refr)) && (depth k MAXDIII)
-), N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(@rand I de e.x + radiance.y + radiance.r) = 0.000

v = true;

- at brdfPdf = EvaluateDiffuse(L, N,) Parmin st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (null)
- andom walk done properly, closely following -/ive)
- ; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, D) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Official:

- Full details in the official assignment 2 document, available today from the website.
- Deadline: <u>May 30th</u>, 23:59.
- Small exhibition of noteworthy entries in a subsequent lecture and on the website.





tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, b) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



Textures

), N); efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(diff if; adiance = SampleLight(&rand, I. e.x + radiance.y + radiance.z) >

v = true;

at brdfPdf = EvaluateDiffuse(L, N st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely for vive)

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR rvive; pdf; i = E * brdf * (dot(N, R) / pdf); sion = true:

Texturing a Plane

 \vec{v}

Given a plane: y = 0 (i.e., with a not Two vectors on the plane le Using these vector We can now u

né plane 🖓

al vector (0,1,0)).

• *P*

eache

leime a color

a ba

 $\vec{u} = (1,0,0)$ and $\vec{v} = (0,0,1)$. $P = \lambda_1 \vec{u} + \lambda_2 \vec{v}.$ $F(\lambda_1, \lambda_2) = \cdots$



tice (depth < NAS

:= inside / i it = nt / nc, dde os2t = 1.8f - ont 0; N); 3)

at a = nt - nc, b = nt = at Tr = 1 - (R0 + (1 fr) R = (D * nnt - N

= diffuse; = true;

-:**fl + refr))** && (depth is MAND)

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I) e.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Factored

indom walk - done properly, closely following -/ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, b) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Ray Tracing
- Intersections
- Shading
- Assignment 2
- Textures



tica ⊾ (depth⊂c NAS

:= inside / : it = nt / nc, dd os2t = 1.0f), N); 3)

at a = nt - nc, b - nt at Tr = 1 - (80 + (1 Tr) R = (0 * nnt - N

E ⁼ diffuse = true;

-:fl + refr)) && (depth is HANDITT

D, N); refl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference) estimation - doing it property if; adiance = SampleLight(&rand, I 2.x + radiance.y + radiance.z) = 0 = 0

v = true; t brdfPdf = EvaluateDiffuse(L, N) = Pauro st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Ond

andom walk - done properly, closely following a /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, U, r pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2017

END OF lecture 3: "Ray Tracing"

Next lecture: "Ray Tracing (2)"

